Multicast Applications of P802.1CQ BARC Address Blocks

v00: 2023-01-17

Roger Marks
(EthAirNet Associates)
roger@ethair.net
+1 802 capable

January 2022

Background

- P802.1CQ ("Multicast and Local Address Assignment") :
  - *specifies protocols, procedures, and management objects for locally-unique assignment of 48-bit and 64-bit addresses in IEEE 802 networks…*
- P802.1CQ/D0.8 uses the **Block Address Registration and Claiming** (BARC) protocol to assign blocks of unicast and multicast addresses

Related documents

- *Automotive Applicability of P802.1CQ Address Assignment*
  - Roger Marks (Aug 2022)
  - https://www.ieee802.org/1/files/public/docs2022/cq-marks-barc-0822-v01.pdf

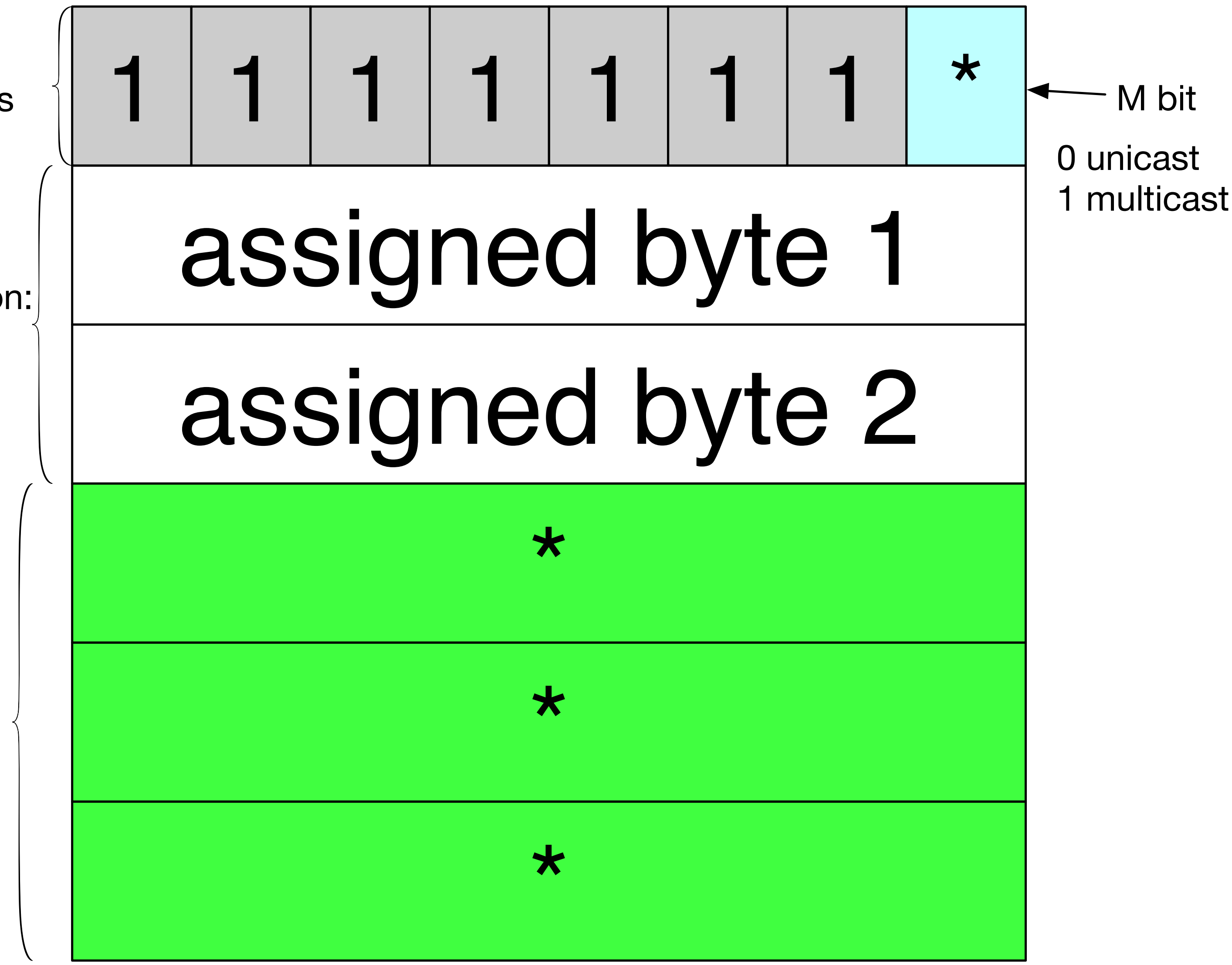
- *P802.1CQ-D0.8, Comment 3 - Supportive Slides*
  - Johannes Specht and Jens Bierschenk (Sep 2022) https://www.ieee802.org/1/files/public/docs2022/cq-specht-d08comment03-0922-v02.pdf

Example of BARC Address Block (AB)

Header:
•Indicates block address format.

Assignment identification:
•Uniquely assigned to station.

Block range:
•Assignable by station.
•Various block sizes supported.



2 contiguous subblocks per AB (one unicast, one multicast)

Why Unicast/Multicast Address Blocks?

TG ballot comment resolution specifies that use case information will be added; e.g.:

- *Include use cases that cover the value of a unified assignment of unicast and multicast address blocks.*

- Device may use multicast addresses as destinations for its streams.
  - set of multicast addresses assigned to a talker, supporting various listener sets
- Block assignment can be exploited in the forwarding process.
  - This can work better when unicast & multicast are assigned together.

Note: The examples here illustrate the Address Blocks, not the protocol for assigning them. Some applications may use the Address Blocks without using BARC (e.g. static assignment), but their use should be consistent with BARC.

Structured Networks

Address blocks offer structured addresses.

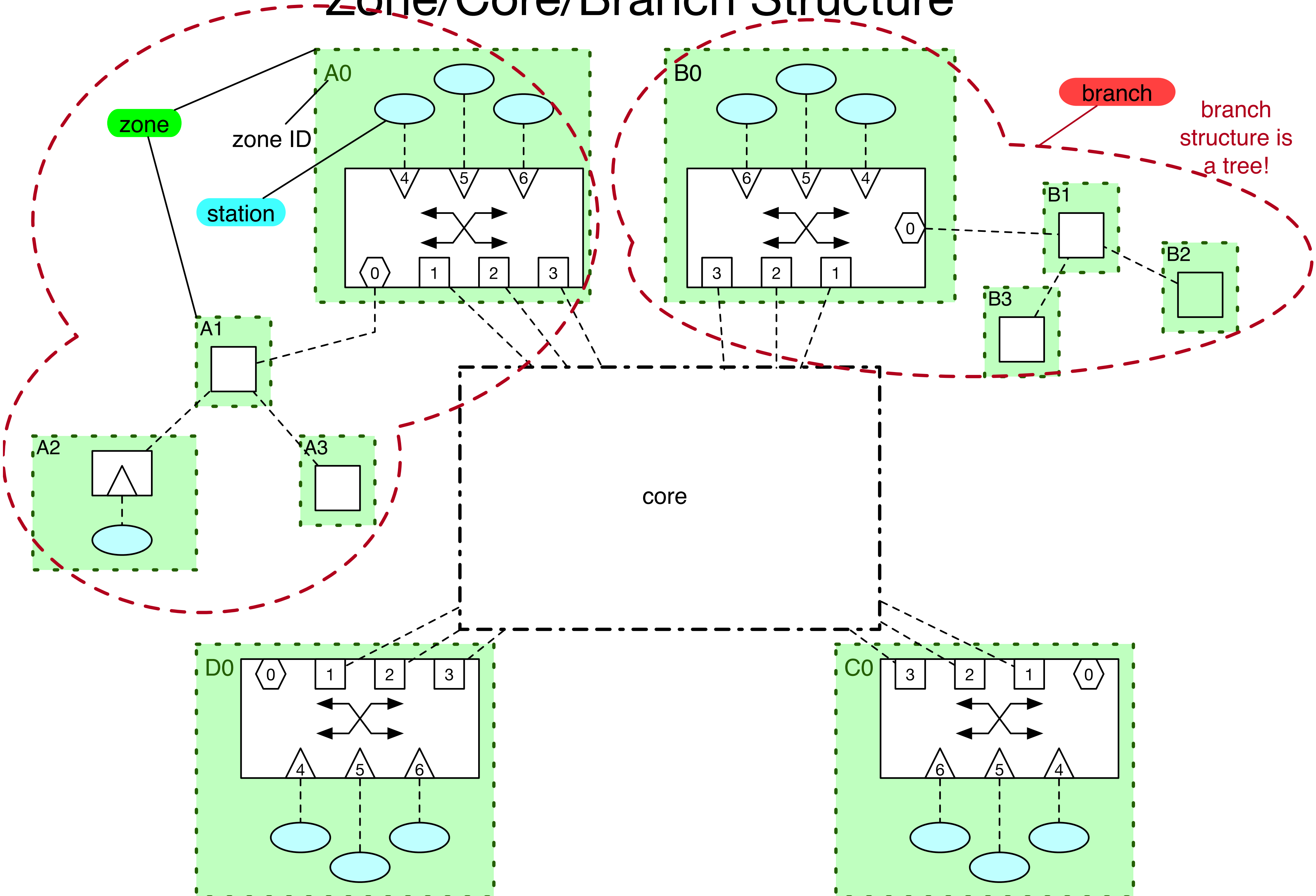
A structured network may suggest how to structure its addresses.

Previous examples have not considered multicast, so they have not helped to explain the utility of joint unicast/multicast address blocks.

Below, I'll introduce an example structure.

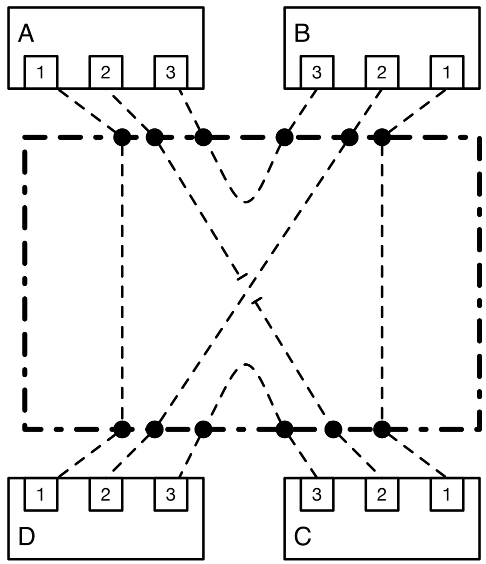
The structure should be simple enough to analyze but complex enough to be both interesting and relevant.

Zone/Core/Branch Structure

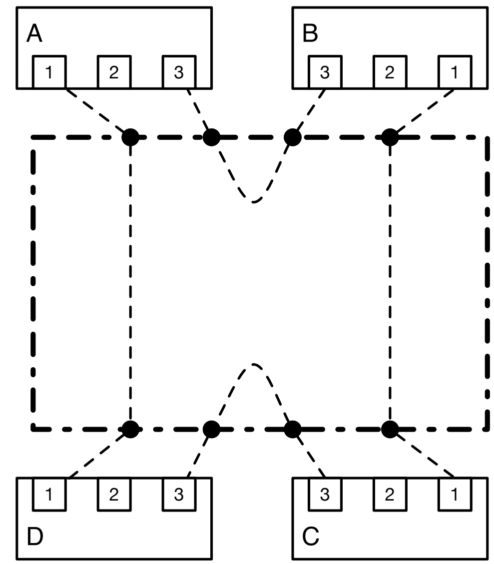


Core: links, not a switched network

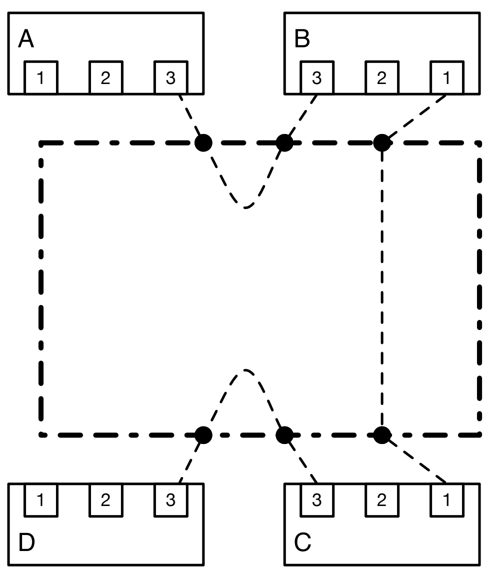
examples:



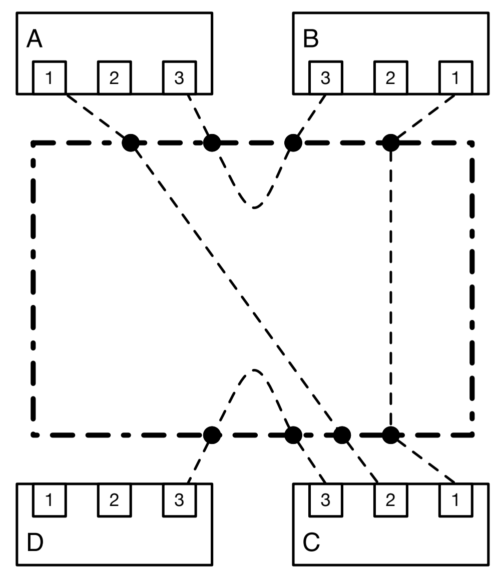
full mesh



loop



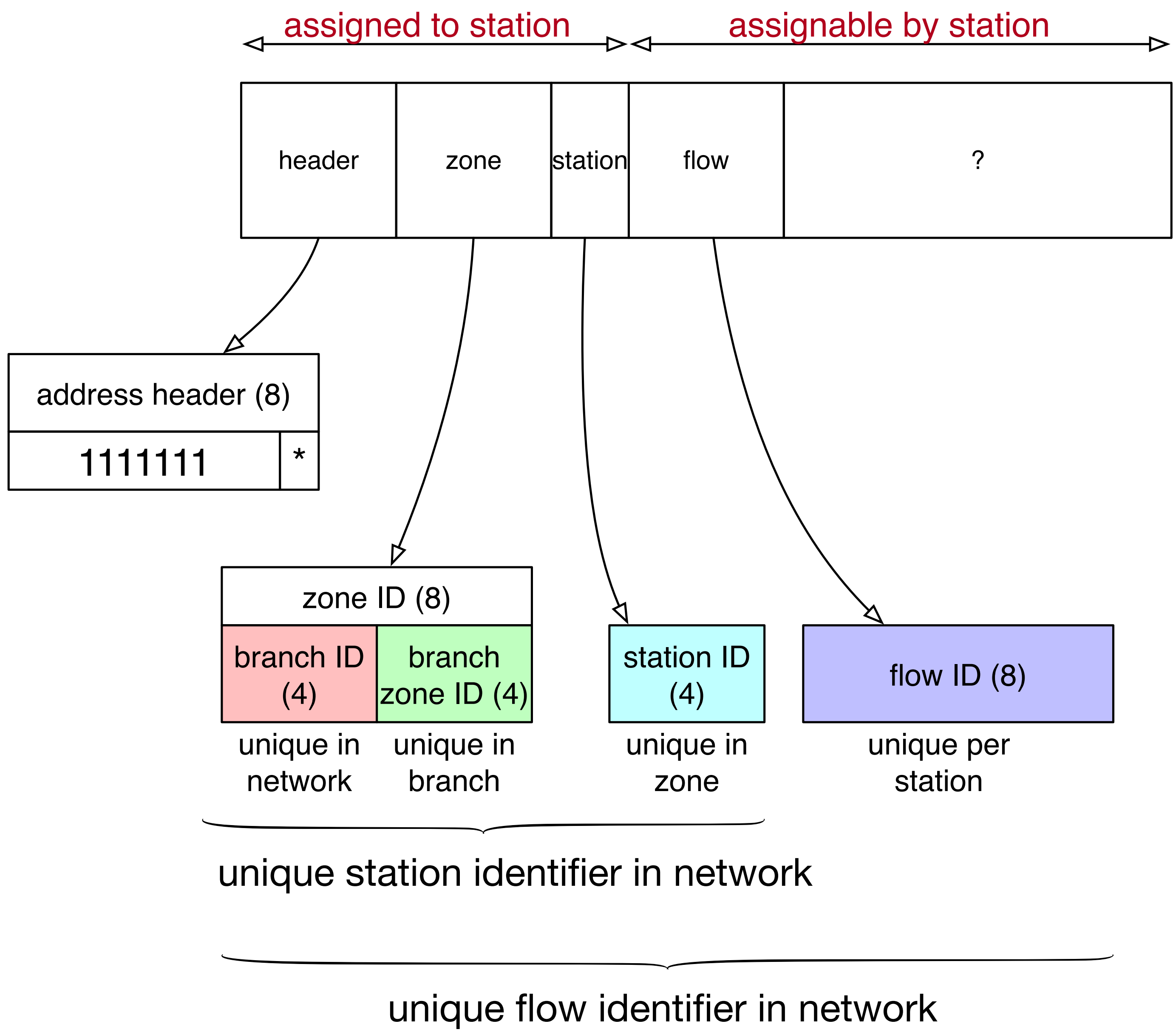
line



loop+
branch

will return to
this topic later

a way to structure station address block assignments



Multicast destinations

assume that listeners notify a talker of their interest in a talker's multicast flow

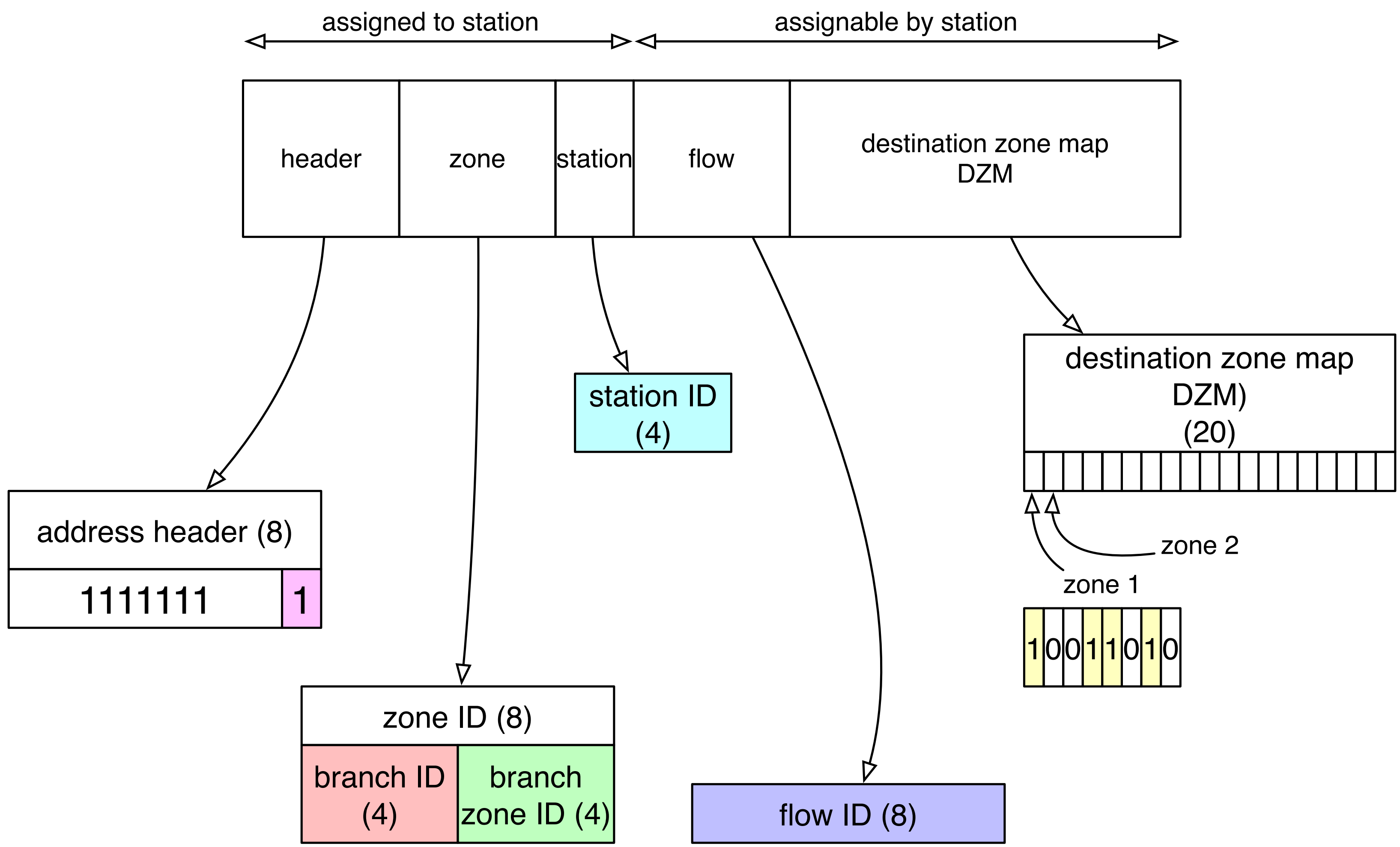
flow ID

zone ID

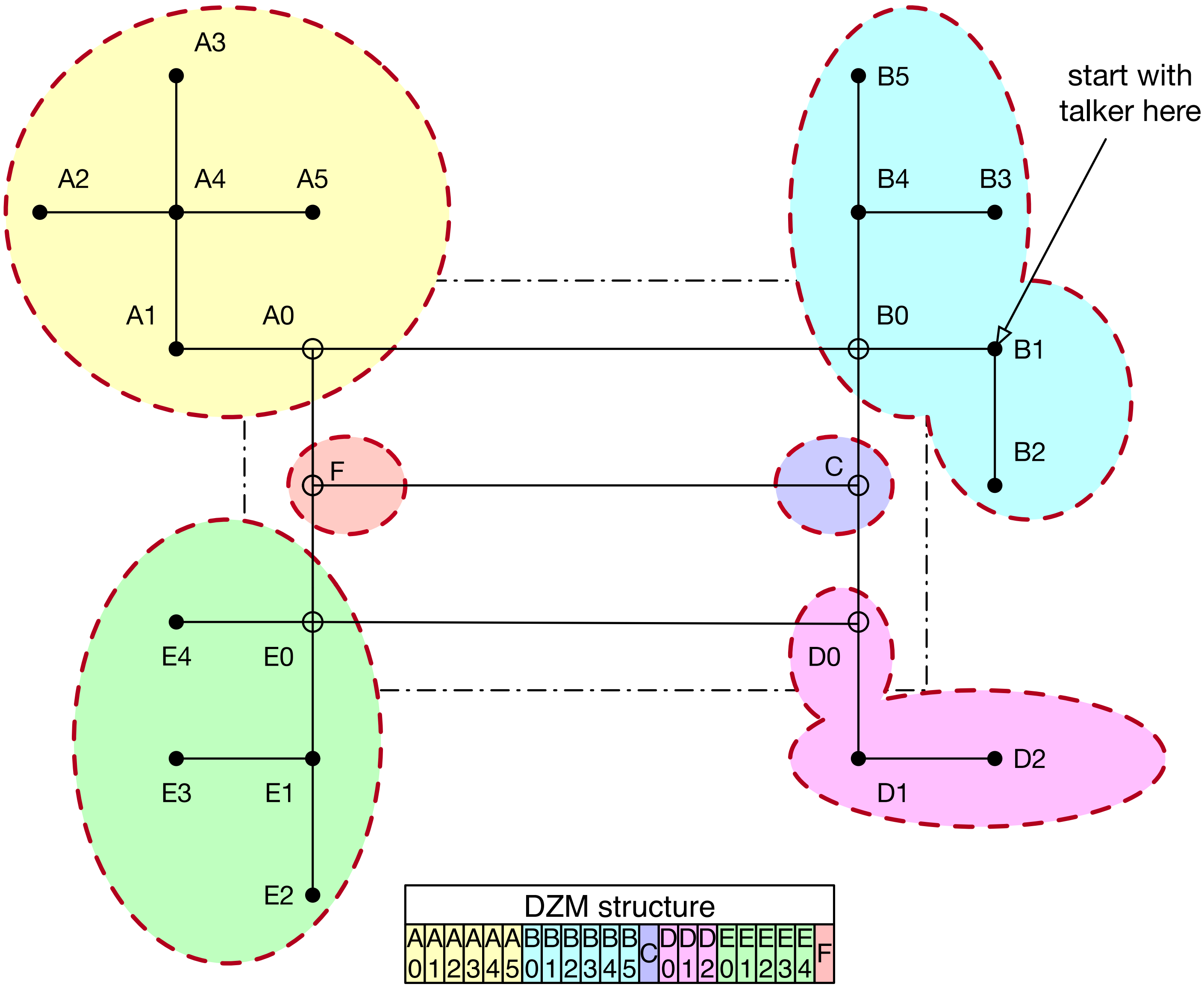
branch ID	branch zone ID	at least one station listener?	Destination Zone Map (DZM)
A	1	yes	1
A	2	no	0
B	1	no	0
B	2	yes	1
B	3	yes	1
C	1	no	0
D	1	yes	1
D	2	no	0

1	0	0	1	1	0	1	0

What is this structure saying to us about <u>multicast</u> address format?



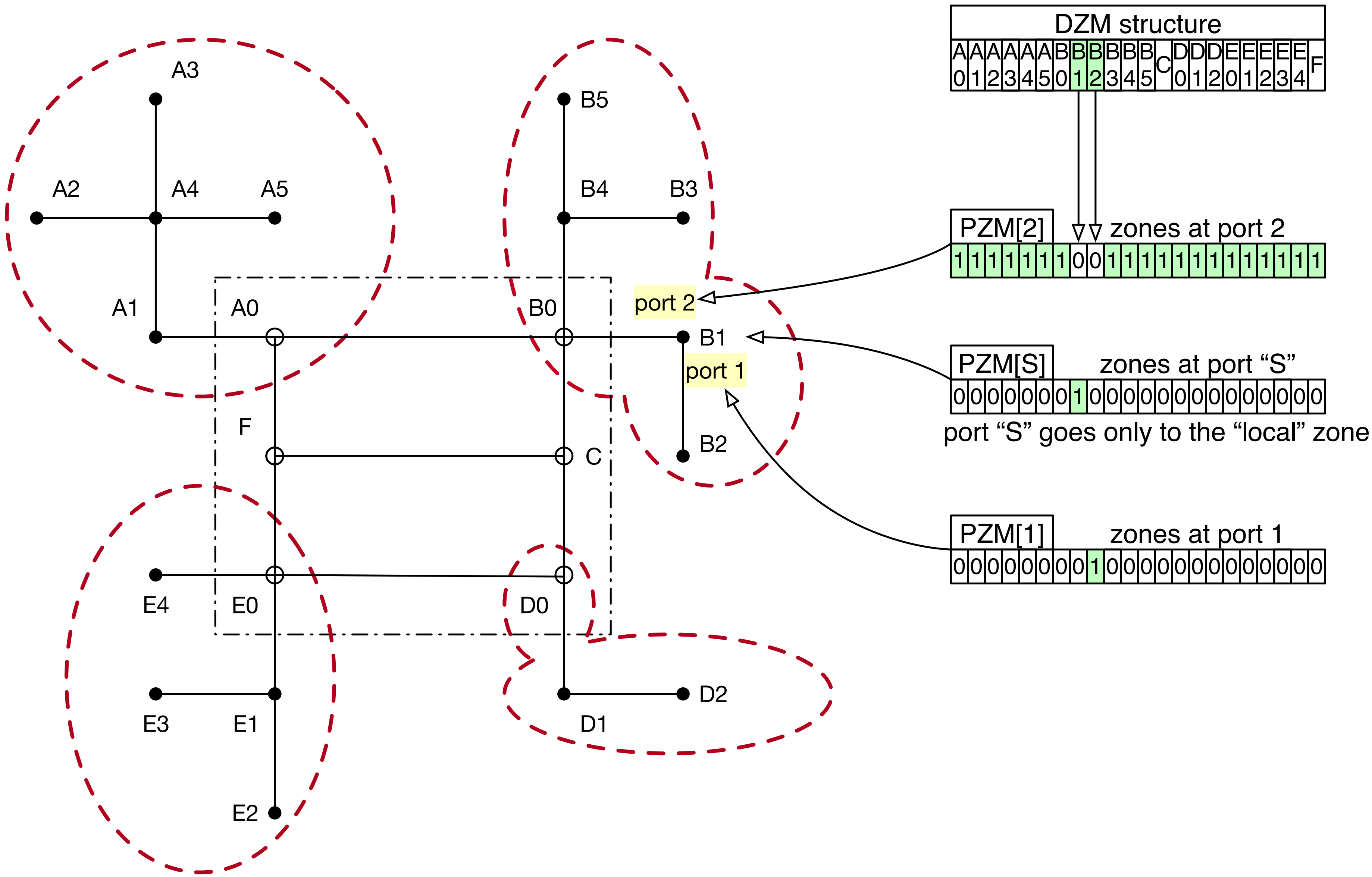
Example network



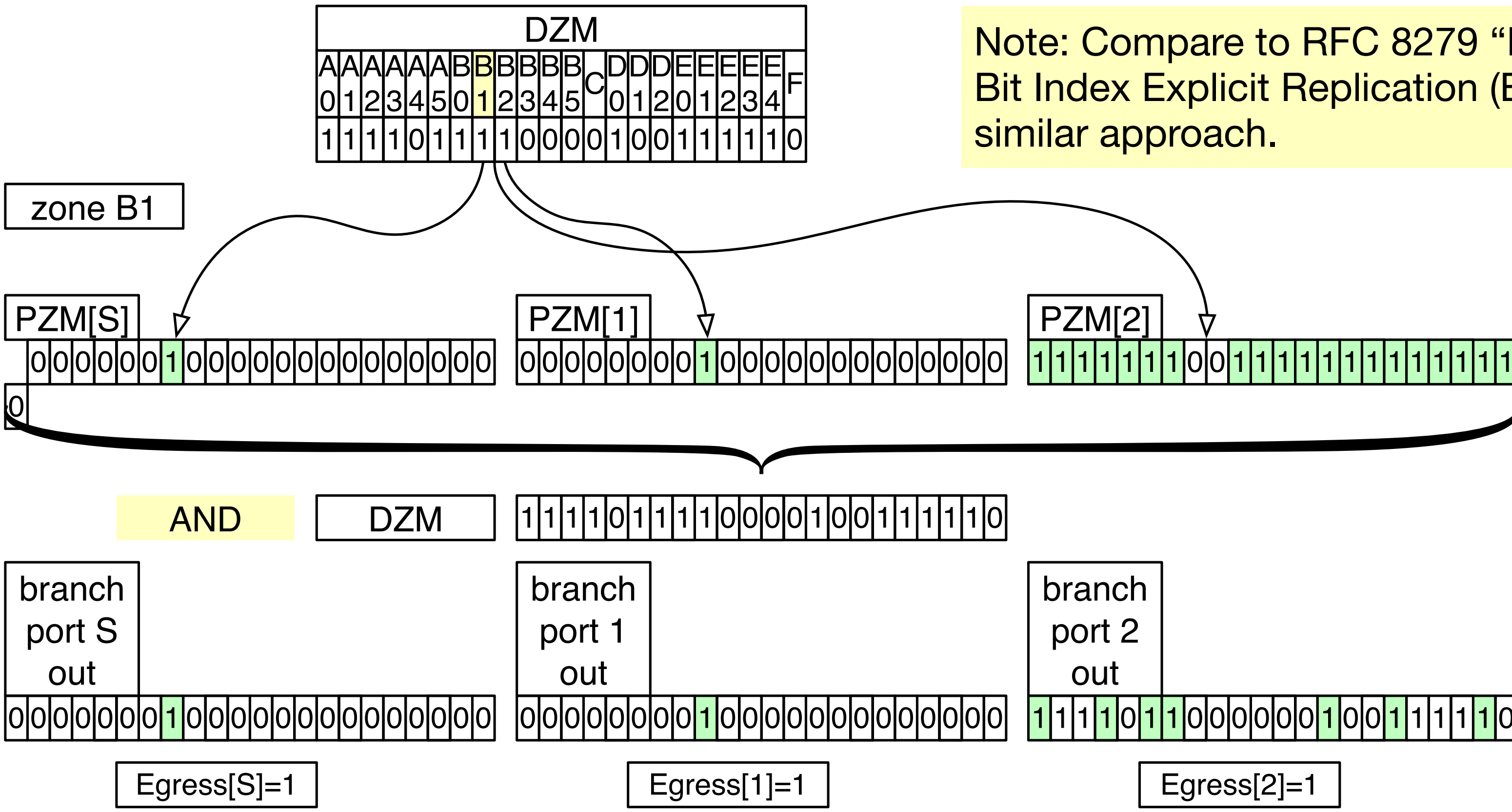
- Just an example.
- However, this is well aligned with network structures I have seen discussed for automotive use.

Port Zone Map (PZM)

maps zones to ports



Forwarding with DZM, in a branch



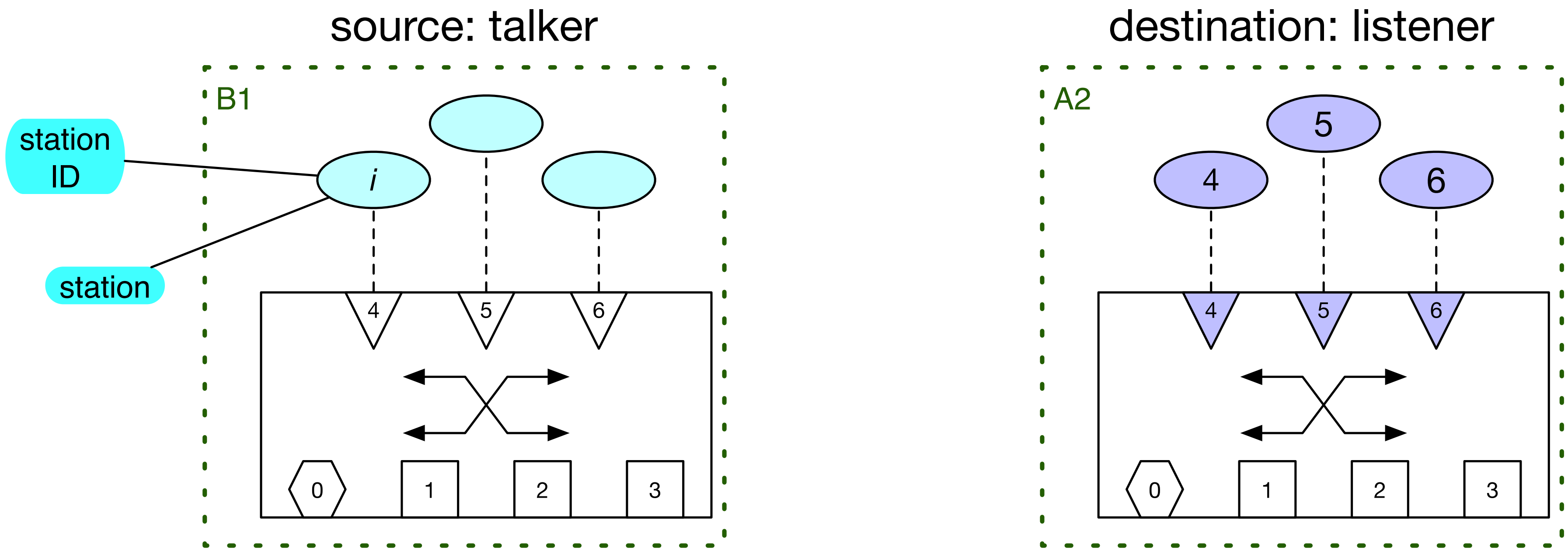
Note: Compare to RFC 8279 “Multicast Using Bit Index Explicit Replication (BIER),” with a similar approach.

- Egress vector: forward at port *n* if, and only if, Egress[*n*]=1
- Egress[*n*] = 1 if, and only if, (PZM[*n*] & DZM) ≠0
  - • i.e. if any destination zone is reachable at port *n*
- within a branch zone, PZM *is* the multicast forwarding database

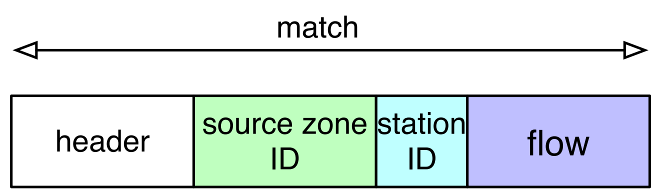
  database size in bits is # of network zones times # of zone ports

  could compress to # of zone ports minus 1, but that hardly seems worthwhile
- What about port S?

In-Zone Forwarding



- station forwarding table lists subscribing listener ports per source flow
  - state held only in local zone
- one entry per flow
- bits per entry (prior example):
  - source zone: 8
  - source station: 4
  - flow: 8
  - plus 1 bit per station port

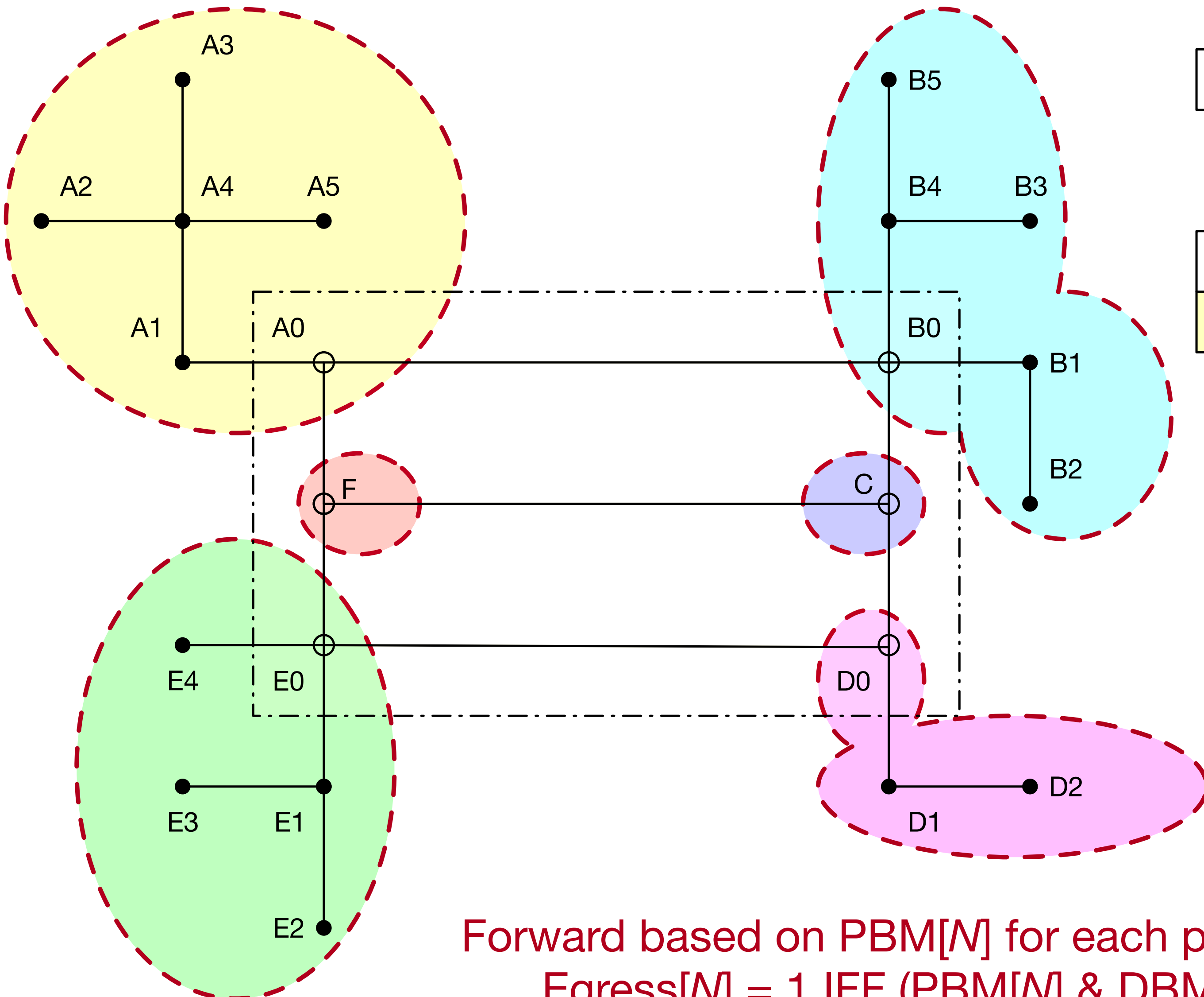


source zone ID	source station ID	flow	Egress[4]	Egress[5]	Egress[6]
A	2	a	1	0	0
A	1	a	0	1	0
A	3	c	1	0	1
B	5	a	1	0	0
C	2	d	1	1	0
C	4	a	1	1	1

And next… the core

forwarding in the core is branch-based

core switching does not need to consider branch zones



destination zone map (DZM)

A0	A1	A2	A3	A4	A5	B0	B1	B2	B3	B4	B5	C	D0	D1	D2	E0	E1	E2	E3	E4	F

destination branch map (DBM)

A	B	C	D	E	F

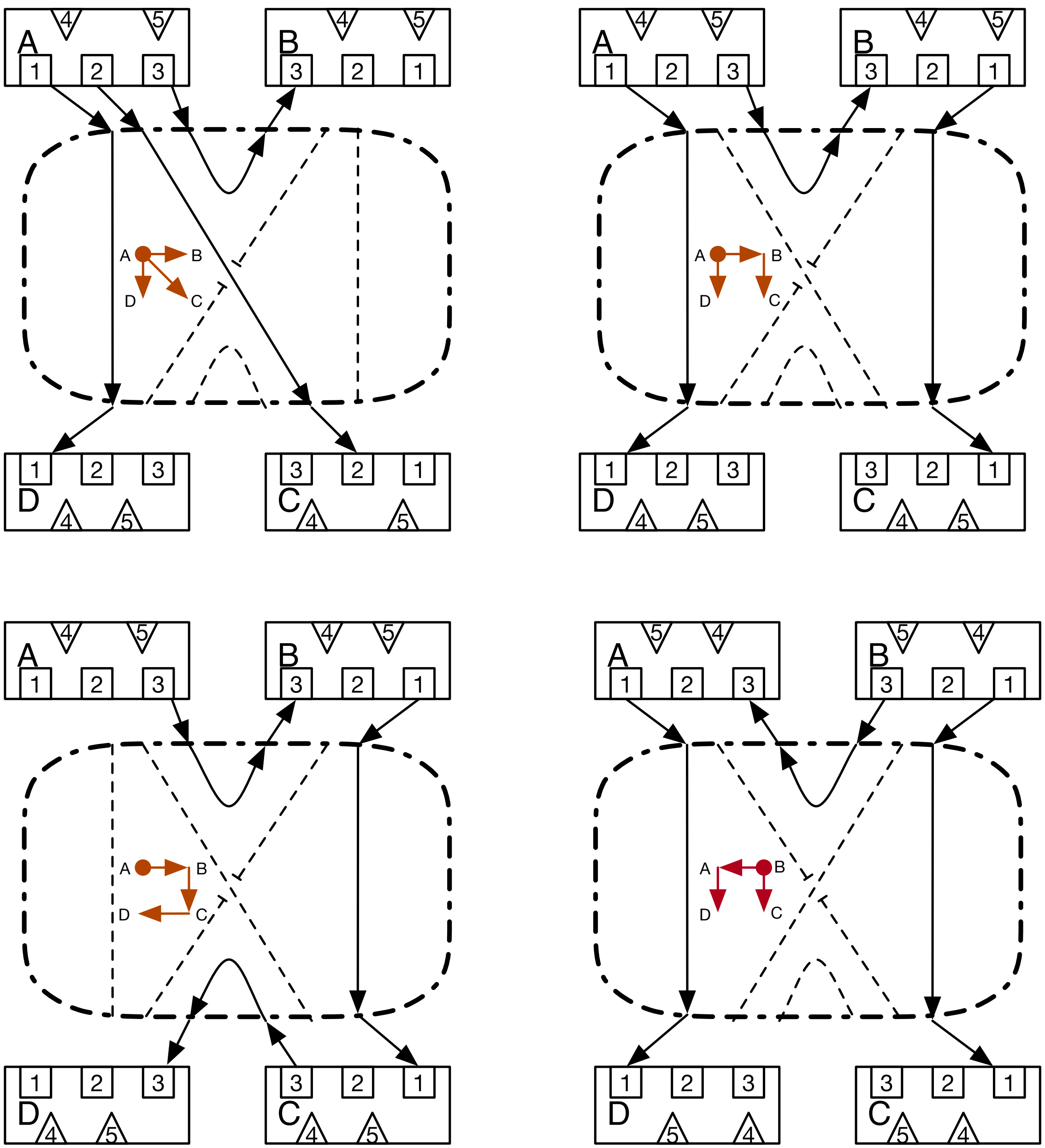
1: at least one destination zone in branch

Forward based on PBM[*N*] for each port:

Egress[*N*] = 1 IFF (PBM[*N*] & DBM) ≠0

But PBM[*N*] is trickier in the (multiply-connected) core

Arborescences

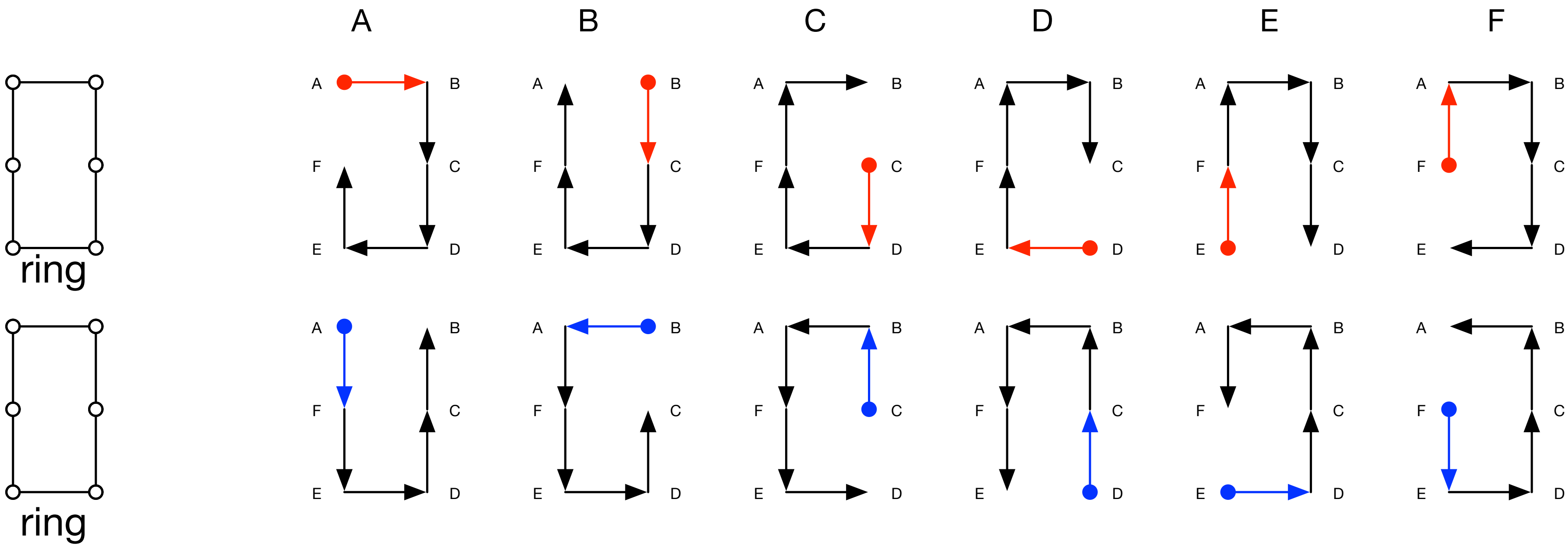


- In graph theory, an arborescence is a directed graph from a root vertex with exactly one path to every node.

- Basically you can pick a vertex and select any tree; the arborescence follows.

- Let’s look at more examples.

Orthogonal Arborescences

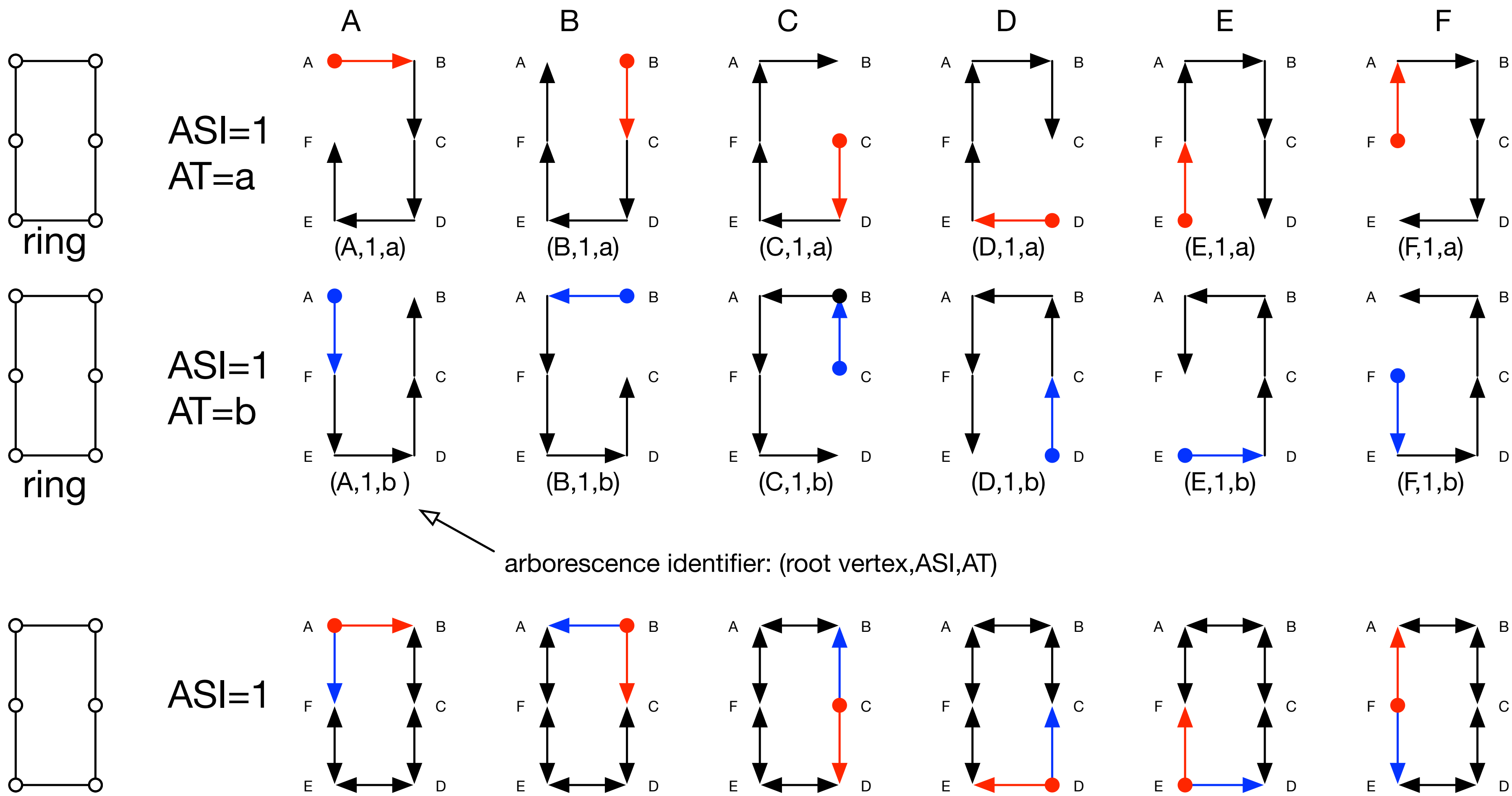


• I've noticed an interested relationship among some arborescences; I call it orthogonality.

• Two arborescences are orthogonal (to each other) unless they share an edge leading into a node and diverge leaving that node [i.e. don't follow the same edge out of that node].

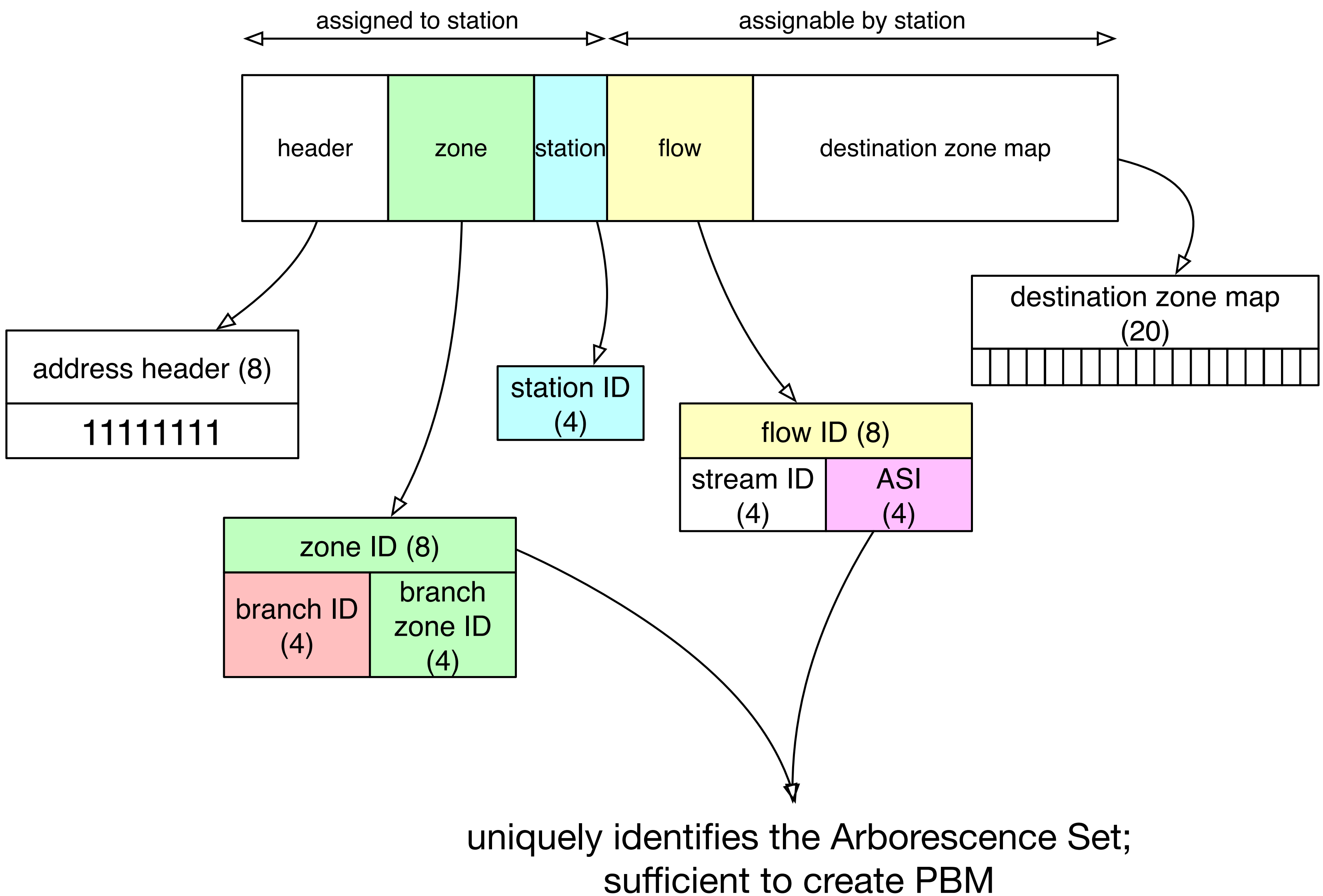
orthogonal non-orthogonal

Arborescence Sets and Arborescence Set ID (ASI)

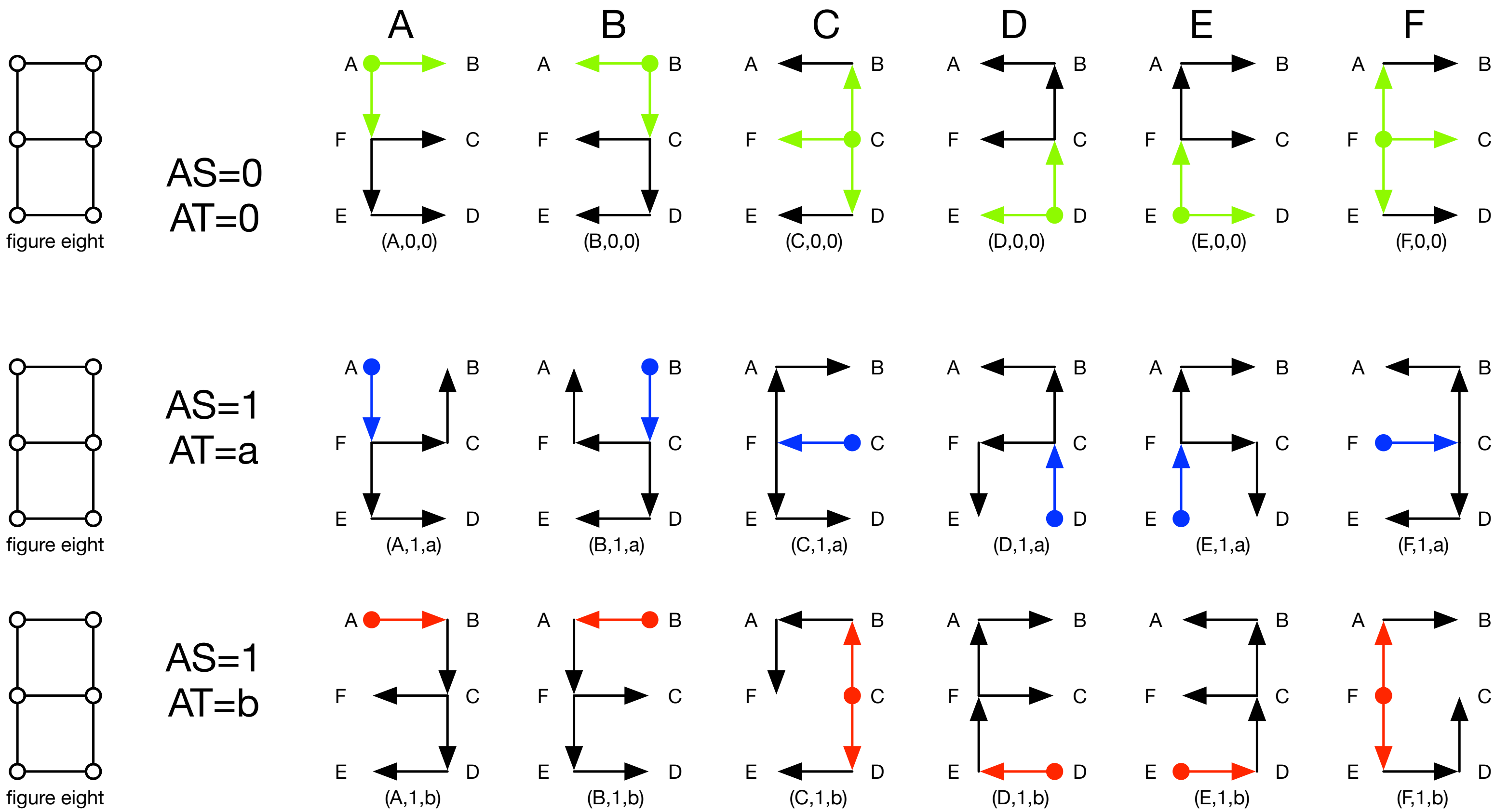


- Arborescence set: set of pairwise-orthogonal arborescences with a common root vertex.
- Arborescence Set Identifier (ASI) can be inserted into the frame; AT is not needed.
- Root vertex chooses the AT, among those in the ASI, by choosing the egress port.
- Root vertex can send duplicate frames (for FRER, etc.) by using multiple arborescences.
  – if they are from the same Set, then ASI is the same and the frame are actual duplicates.
    -i.e. do not need to generate two different frames

Multicast address format with Arborescence Set Identifier (ASI)



Arborescence Set Examples: Figure Eight



- An egress port supports only one arborescence per AS.
- In the Figure Eight, root vertex A, B, D, and E have two ports and support up to two arborescences per set.
  - see AS=1, with AT=a and AT=b.
  - but AS=0 consumes both ports, so the set includes only one.
- Also: notice the property of the AS that a failed link will leave at least one path intact to each branch. Non-orthogonal arborescences never have this feature.
  - Good basis for FRER: send frame on both of the orthogonal arborescences.

Selecting the arborescence

The core zone receiving a frame from a branch or station assigns the AS:

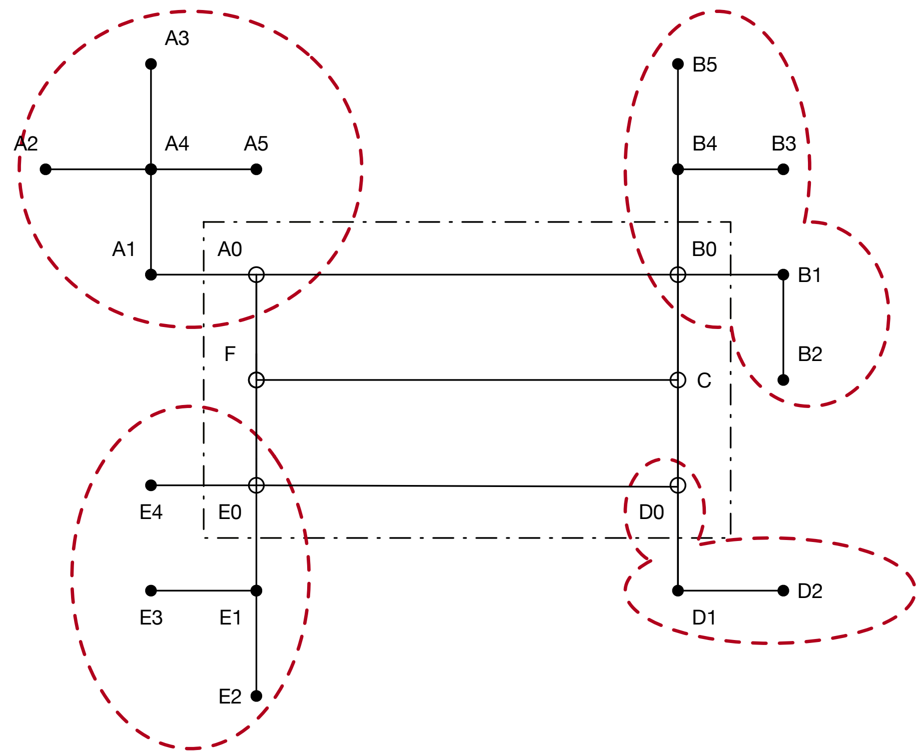
– inserts AS into the frame*

– used the egress port(s) for the selected arborescence(s)

A stream can be mapped to a single arborescence, to avoid misordering.

*Note: This is the only case of frame editing required. And it's only one edit: every copy of the emitted frame is identical.

Port assignment tables are simple:



Zone C

ASI	AT	Egress[1]	Egress[2]	Egress[3]
0	0	1	1	1
1	a	0	0	1
1	b	1	1	0
2	a	0	0	1
2	b	0	1	0
2	c	1	0	0

Zone B0

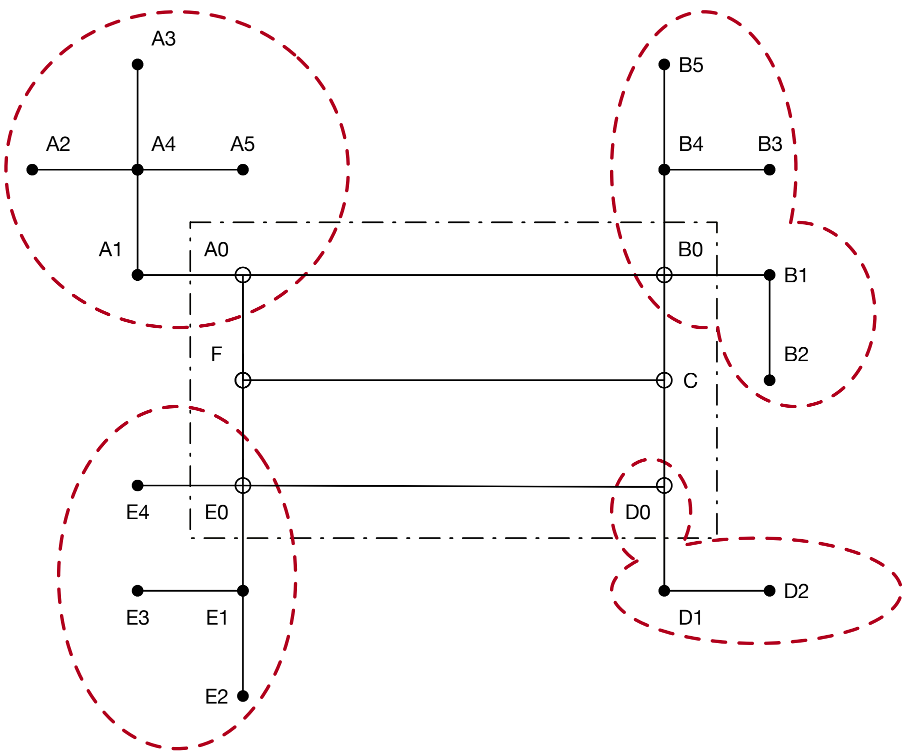
ASI	AT	Egress[1]	Egress[2]
0	0	1	1
1	a	1	0
1	b	0	1

Forwarding a frame with an ASI

Forwarding a frame with a labeled ASI inside the core involves:

(1) For each port *N*, look up PBM[*N*] in arborescence port table, based on ASI, source branch ID, and ingress port

(2) Egress[*N*] = 1 IFF (PBM[*N*] & DBM) ≠0
(as explained earlier)

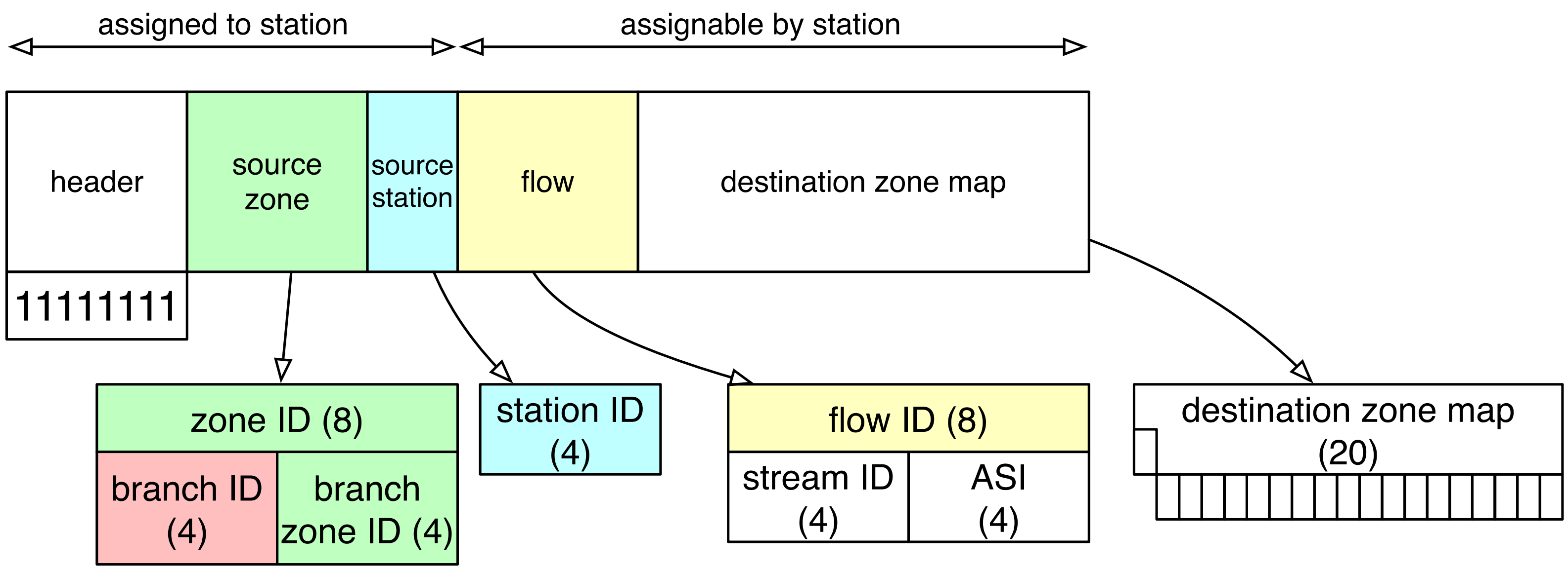


Zone C arborescence port table

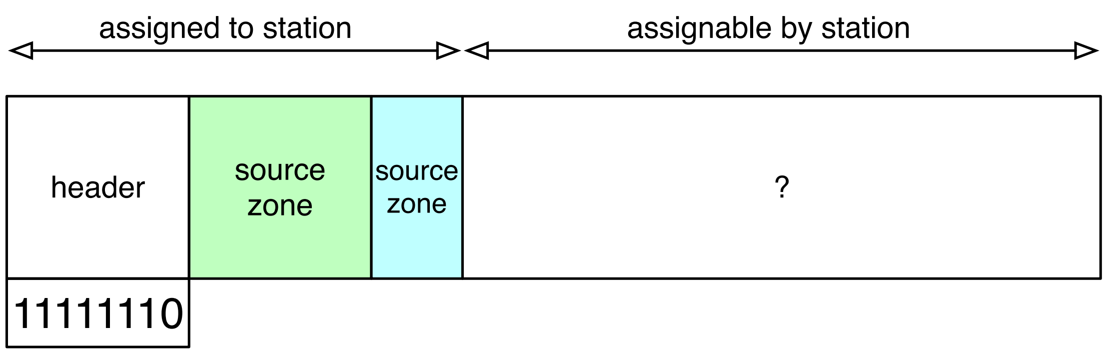
ASI	branch ID		PBM[1]	PBM[2]	PBM[3]
0	A		–	–	–
0	B		–	000110	000001
0	D		110000	–	000001
0	E		–	–	–
0	F		–	–	–
1	A	ingress port 1	–	000110	000001
1	B			000110	100001
1	D	ingress port 2	110000	–	000011
1	E		110000		000001
1	F		–		–
1	A	ingress port 3	010000	–	–
1	B		–	–	
1	D		–	–	
1	E		–	000100	
1	F		110000	000110	

Source Address

Destination Address field



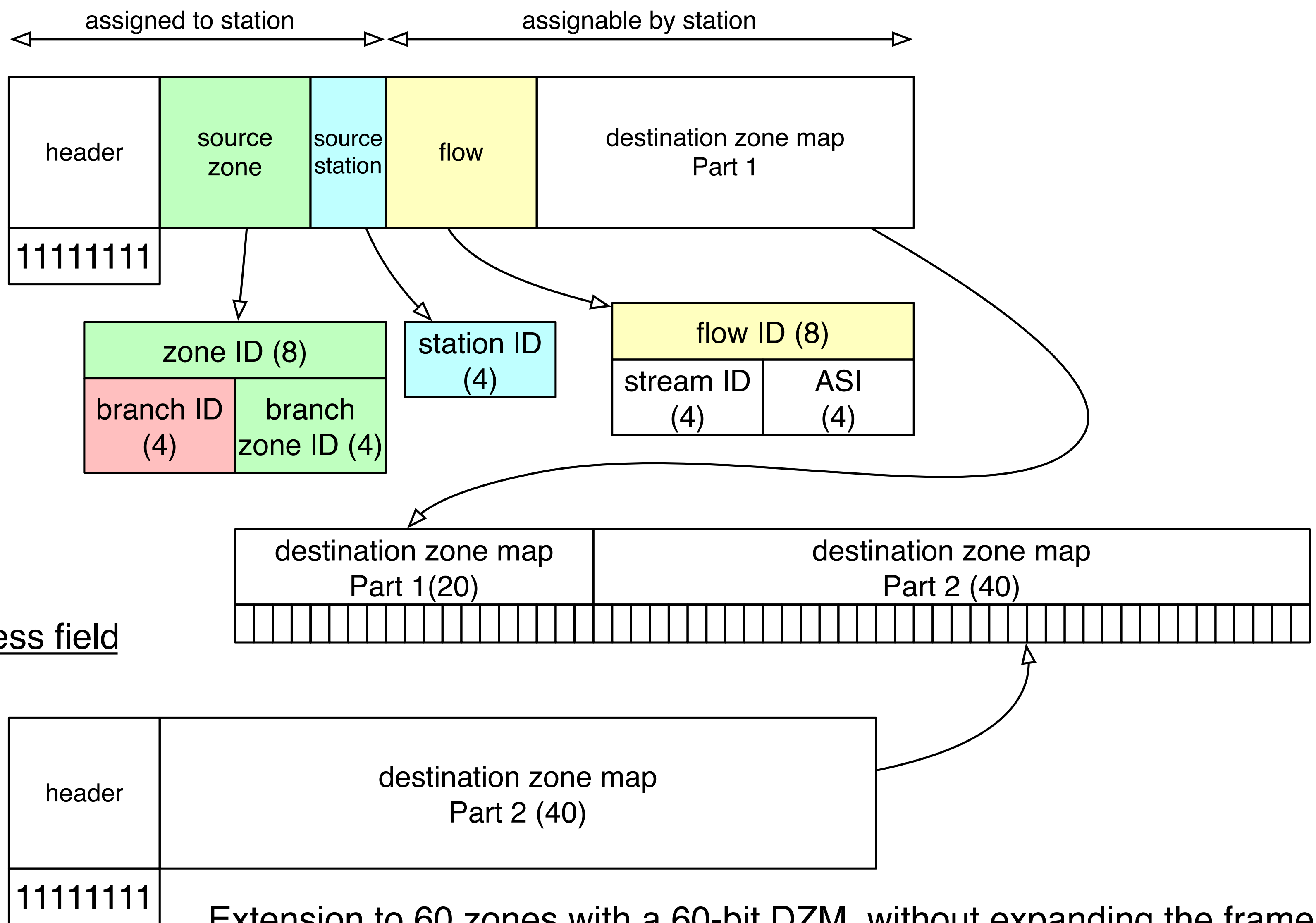
Source Address field



What use is the source address? The fields are all redundant! The source ID is in the DA.

Expanding the DZM, adding zones

Destination Address field



Source Address field

Extension to 60 zones with a 60-bit DZM, without expanding the frame.

Using a multicast address in the SA field, to indicate “no learning”.

Scaling the example

The example configuration supports:

- 6 core zones
- 6 branches
- 9 non-core zones per branch (60 zones total)
  - branch connectivity using arbitrary tree
- 16 stations per zone (960 stations total)
- 16 streams per station (15,360 streams)
- 16 arborescence sets per branch
  - 2 or 3 arborescences per set

Forwarding table storage requirements – core zone

STORAGE
arborescence port table, maximum scale:
# of entries: #of arbs * (# of branches – 1)
In the example: three arbs, 6 branches: 15 entries

Each entry is:
- ASI (4 bits)
- branch ID (4 bits)
- port ID (assume 4 bits)
- port count [3] * branch count [6]
- TOTAL: 30 bits
  - –but much less, since table is sparse
  - –in this table, 29 of 45 entries are empty

Full table: <450 bits [but scales with # of arbs]
- ~256 bits in the table to the right
- independent of # of zones, stations, and streams!

In addition: zone-to-branch mapping storage
- 1 per port (3 ports)
- 1 bit per zone (60 zones)
- TOTAL: 180 bits

CALCULATIONS:
Computing DBM:
- N bitwise AND operations for N ports (N=3 here)
- each operation uses I bits for I zones (assume 60)

Computing Egress vector
- no more than 2 bitwise AND operations
- each operation uses K bits (K=#of branches=6)

Zone C arborescence port table

<table>
<tr><th>ASI</th><th>branch ID</th><th></th><th>PBM[1]</th><th>PBM[2]</th><th>PBM[3]</th></tr>
<tr><td>0</td><td>A</td><td></td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>0</td><td>B</td><td></td><td>–</td><td>000110</td><td>000001</td></tr>
<tr><td>0</td><td>D</td><td></td><td>110000</td><td>–</td><td>000001</td></tr>
<tr><td>0</td><td>E</td><td></td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>0</td><td>F</td><td></td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>1</td><td>A</td><td rowspan="2">ingress port 1</td><td rowspan="2">–</td><td>000110</td><td>000001</td></tr>
<tr><td>1</td><td>B</td><td>000110</td><td>100001</td></tr>
<tr><td>1</td><td>D</td><td rowspan="3">ingress port 2</td><td>110000</td><td rowspan="3">–</td><td>000011</td></tr>
<tr><td>1</td><td>E</td><td>110000</td><td>000001</td></tr>
<tr><td>1</td><td>F</td><td>–</td><td>–</td></tr>
<tr><td>1</td><td>A</td><td rowspan="5">ingress port 3</td><td>010000</td><td>–</td><td rowspan="5">–</td></tr>
<tr><td>1</td><td>B</td><td>–</td><td>–</td></tr>
<tr><td>1</td><td>D</td><td>–</td><td>–</td></tr>
<tr><td>1</td><td>E</td><td>–</td><td>000100</td></tr>
<tr><td>1</td><td>F</td><td>110000</td><td>000110</td></tr>
</table>

Unicast Forwarding

- The multicast forwarding scenario can easily incorporate unicast.
  - could simply multicast with a single destination
  - could easily achieve the same result without carrying a DZM

Summary

- Detailed use case shows the application of BARC Address Blocks in a multicast scenario.
- BARC structured Address Blocks provide the possibility of low-complexity forwarding in a structured network.
- Such Address Blocks could be assigned by BARC protocols.
  - static configuration could also be used
  - statically-assigned Address Blocks should follow the BARC semantics (e.g. header field)

Call for Participation

- I ask for you participation in summarizing key points of this use case, and others to meet other needs, in a simplified form for use in IEEE P802.1CQ draft.

- I request review of future P802.1CQ drafts and comments to align the content with practical use cases.